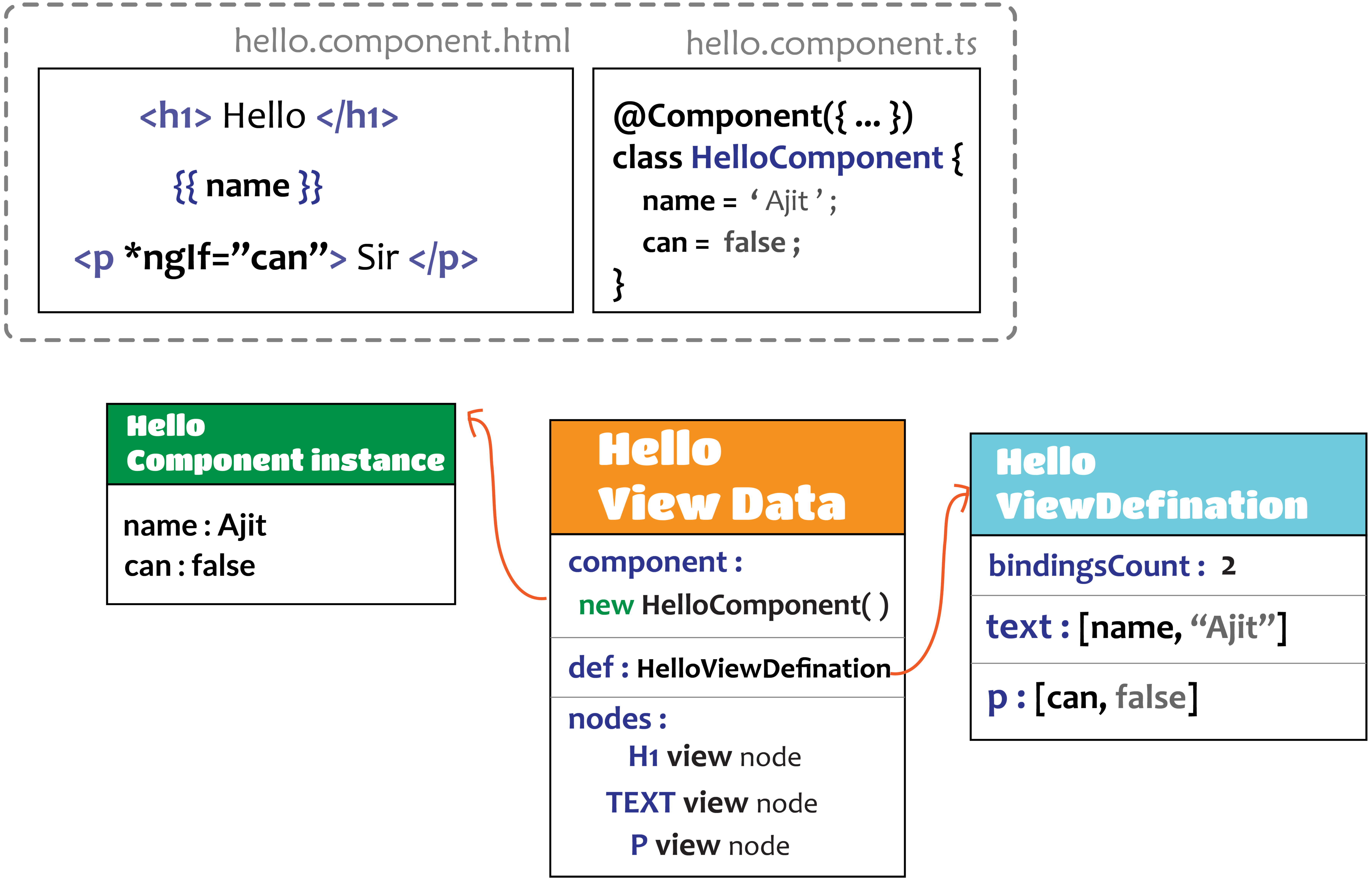


hello.component.html

```html
<h1> Hello </h1>

{{ name }}

<p *ngIf=”can”> Sir </p>
```

hello.component.ts

```
@Component({ ... })
class HelloComponent {
   name = ‘ Ajit ’ ;
   can =  false ;
}
```

Hello Component instance

name : Ajit
can : false

Hello View Data

component :
 new HelloComponent( )

def : HelloViewDefination

nodes :
 H1 view node
 TEXT view node
 P view node

Hello ViewDefination

bindingsCount : 2

text : [name, “Ajit”]

p : [can, false]

internal conversion/representation

SLIDE 1 - Component View Defination & Data

hello.component.html

```
<h1> Hello </h1>
{{ name }}
<p *ngIf="can"> Sir </p>
```

hello.component.ts

```
@Component({ ... })
class HelloComponent {
  name = ' Ajit ';
  can = false ;
}
```

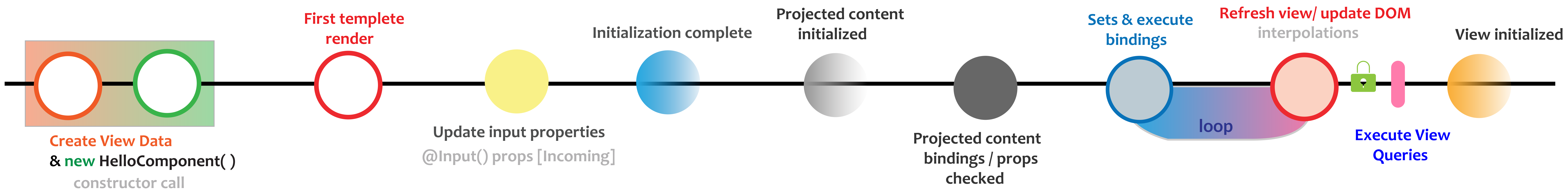
Hello View Data

instance (component instance ref)
bindingsCount : 2
name (binding)
can (binding)

Hello Component instance

name : Vinay ~~Ajit~~
can : false

(class instance)



Hello View State (LOCKED)

name: Ajit
can: false

Hello Execute / Verify Changes

name: Vinay ❌
can: false

MAIN STACK

- Verification Loop
- Life cycle process completed
- this.name = Vinay
- View Initialized
- Life cycle started

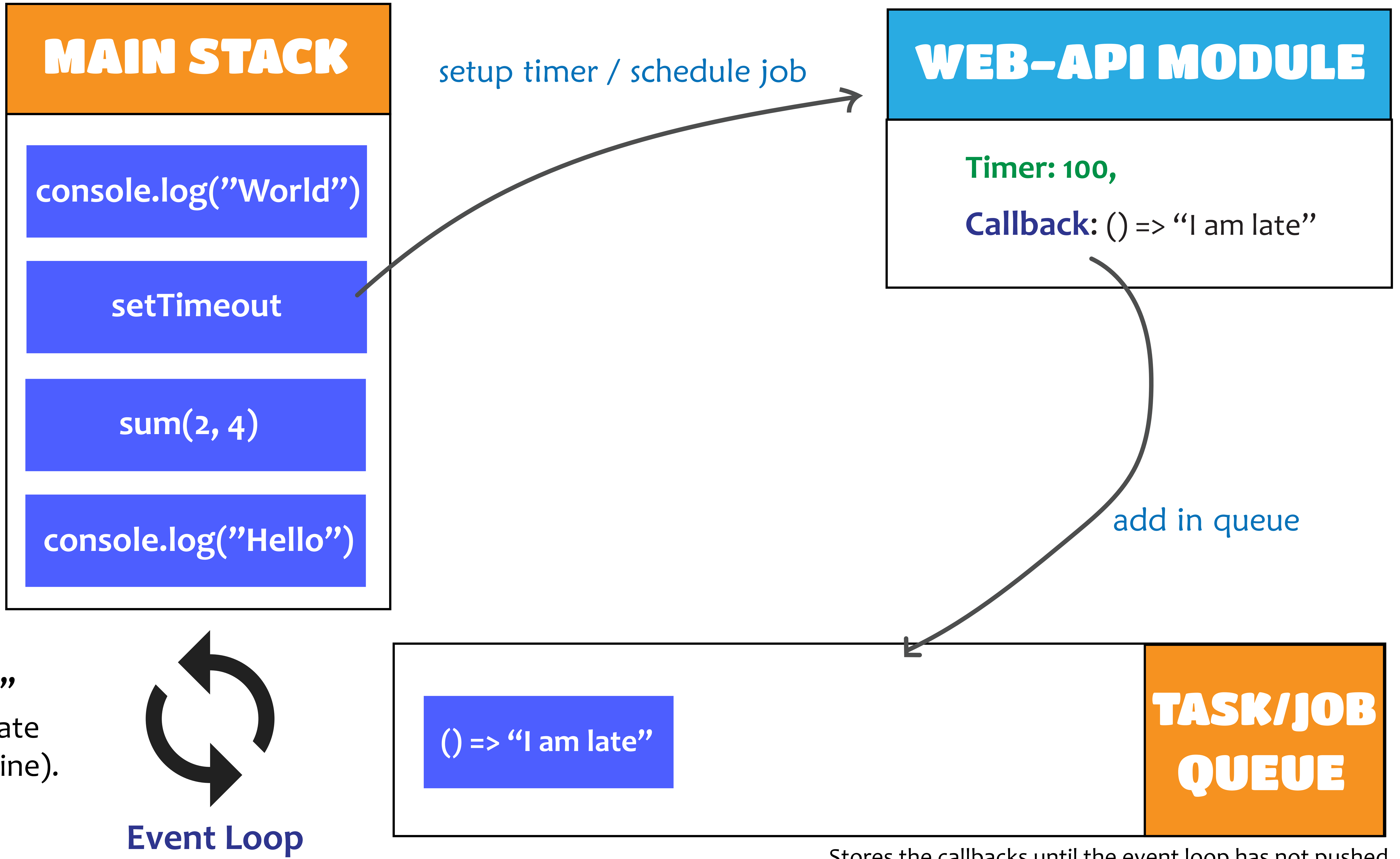
Viewport

Hello

Vinay

SLIDE 2 - Component Life Cycle & Verification loop

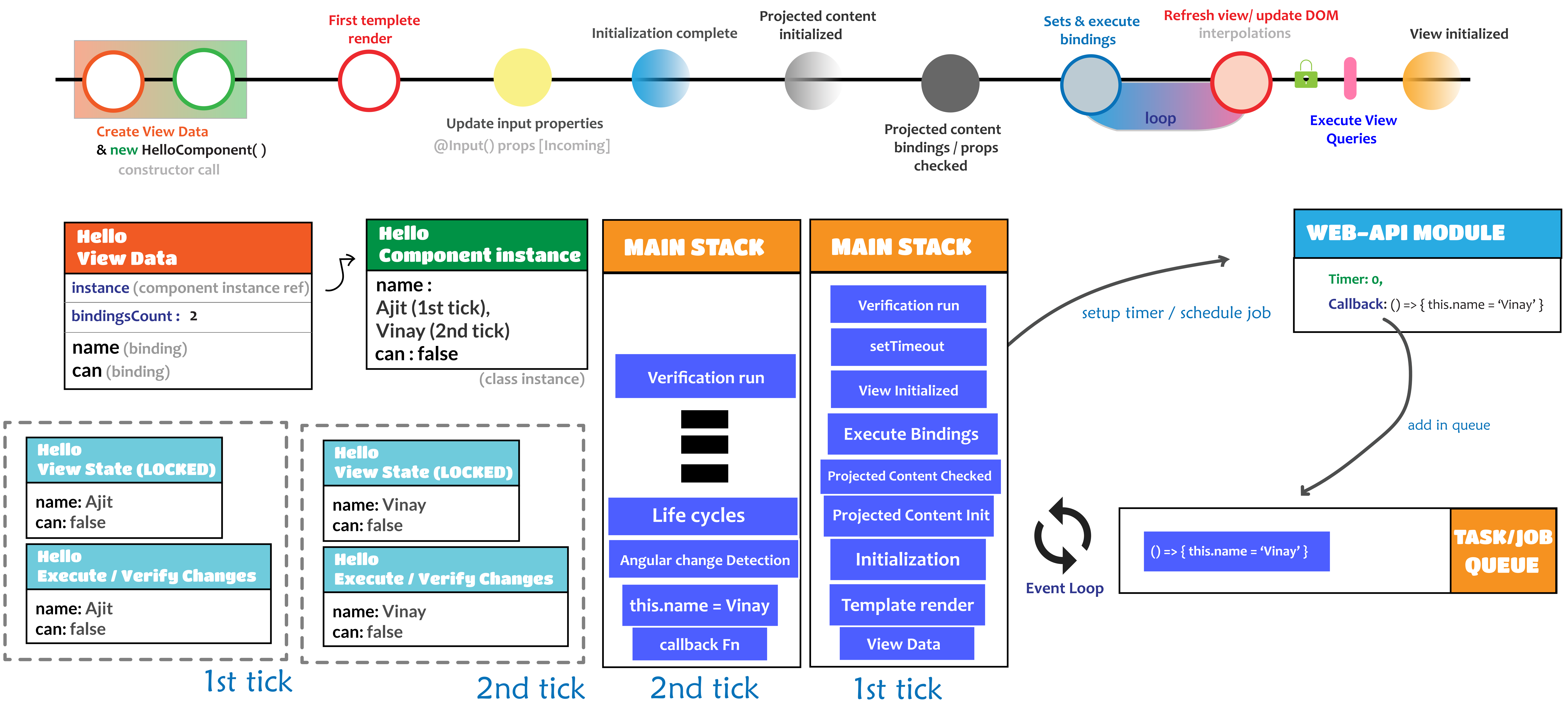
```
console.log("Hello");

sum(2, 4);

setTimeout(() => "I am late", 100);

console.log("World");
```



One complete cycle of "**continuous executions**" in the main stack from its **filled** state to **empty** state is called "One tick" of Javascript VM (Virtual machine).

Stores the callbacks until the event loop has not pushed them into the main stack **(until the main stack is free)**

When stack is empty/free (1 tick completed)
Check queue
Place in stack/execute (2nd tick)



SLIDE 4 - Event Loop Tick & Component Lifecycle

hello.component.html

```
<h1> Hello </h1>
{{ name }}
<p *ngIf="can"> Sir </p>
<greet [message]="message"> </greet>
```

hello.component.ts

```
@Component({ ... })
class HelloComponent {
  name = ' Ajit ' ;
  can = false ;
  message = 'Morning';
}
```

Hello View Data

instance (component instance ref)

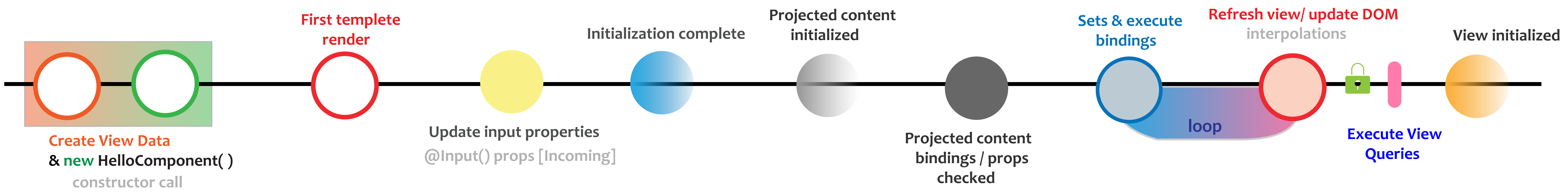
bindingsCount : 3

name (binding)
can (binding)
message (binding)

Hello Component instance

name : Ajit
can : false
message : Good Morning ~~Morning~~

(class instance)



Hello View State (LOCKED)

name: Ajit
can: false
message: Morning

Hello Execute / Verify Changes

name: Ajit
can: false
message: Good Morning ❌

MAIN STACK

Verification Loop

Life cycle process completed

this.message = 'Good Morning'

View Initialized

Life cycle started

Viewport

Hello

Ajit

Greet with Good Morning

hello.component.html

```html
<h1> Hello </h1>
 {{ name }}
<p *ngIf="greetCmp"> For you </p>
<app-greet> </app-greet>
```

hello.component.ts

```typescript
@Component({ ... })
class HelloComponent {
   @ViewChild(GreetComponent)
   greetCmp: GreetComponent;
   name = ' Ajit ' ;
}
```

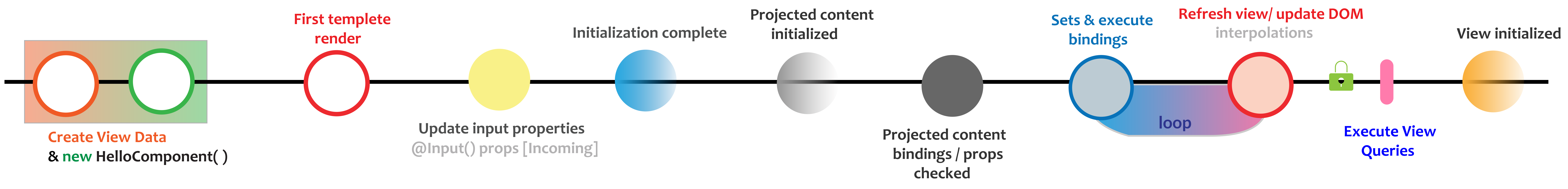
Hello View Data

instance (component instance ref)
bindingsCount : 2
name (binding)
greetCmp (binding)
app-greet (component)

Hello Component instance

greetCmp : GreetCmp ~~undefined~~
name : Ajit

(class instance)



Hello View State (LOCKED)

name: Ajit
greetCmp: undefined

Hello Execute / Verify Changes

name: Ajit
greetCmp: GreetComp instance ❌

greetCmp instance gets defined

MAIN STACK

Verification Loop
Life cycle process completed
Execute view queries
View Initialized
Life cycle started

Viewport

Hello
Ajit
For you
Flowers

SLIDE 6 - @View() Child Query Binding

hello.component.html

```html
<h1> Hello </h1>
 {{ name }}
<p *ngIf="greetCmp"> For you </p>
<app-greet> </app-greet>
```

hello.component.ts

```typescript
@Component({ ... })
class HelloComponent {
  @ViewChild(GreetComponent, { static: true })
   greetCmp: GreetComponent;
   name = ' Ajit ';
}
```

Hello View Data

instance (component instance ref)

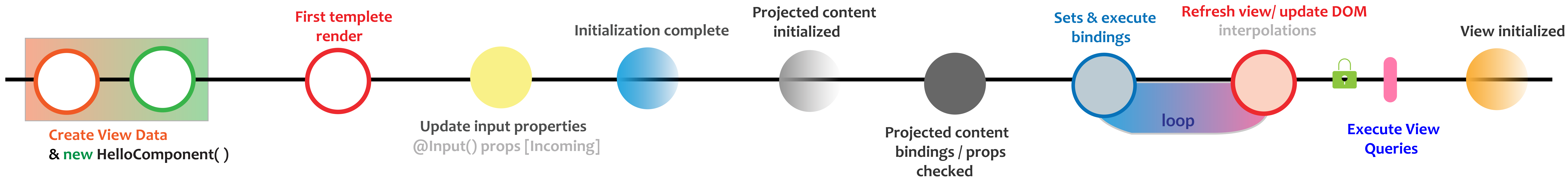
bindingsCount : 2

name (binding)
greetCmp (binding)
app-greet (component)

Hello Component instance

greetCmp : GreetCmp (static resolve)
name : Ajit

(class instance)



Hello View State (LOCKED)

name: Ajit
greetCmp: GreetComp instance

Hello Execute / Verify Changes

name: Ajit
greetCmp: GreetComp instance

MAIN STACK

- Verification Loop
- Life cycle process completed
- Static resolve child queries
- Template render
- View Data
- Life cycle started

Viewport

Hello

Ajit

For you

Flowers

SLIDE 7 - Static Option & @View() Child Query Binding

hello.component.html

```
<h1> Hello </h1>
{{ name }}
<p *ngIf=”greetCmp.isAllowed”> For you </p>
<app-greet> </app-greet>
```

hello.component.ts

```
@Component({ ... })
class HelloComponent {
  @ViewChild(GreetComponent)
   greetCmp: GreetComponent;
   name = ‘ Ajit ’ ;
}
```

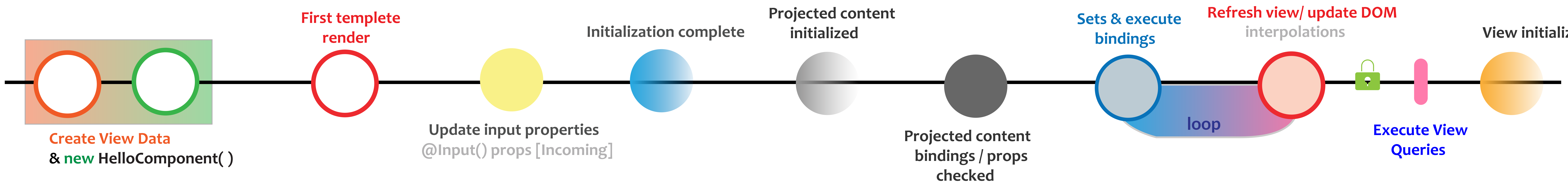
Hello View Data

instance (component instance ref)
bindingsCount : 2
name (binding)
greetCmp.isAllowed (binding)
app-greet (component)

Hello Component instance

greetCmp : GreetCmp ~~undefined~~
name : Ajit

(class instance)

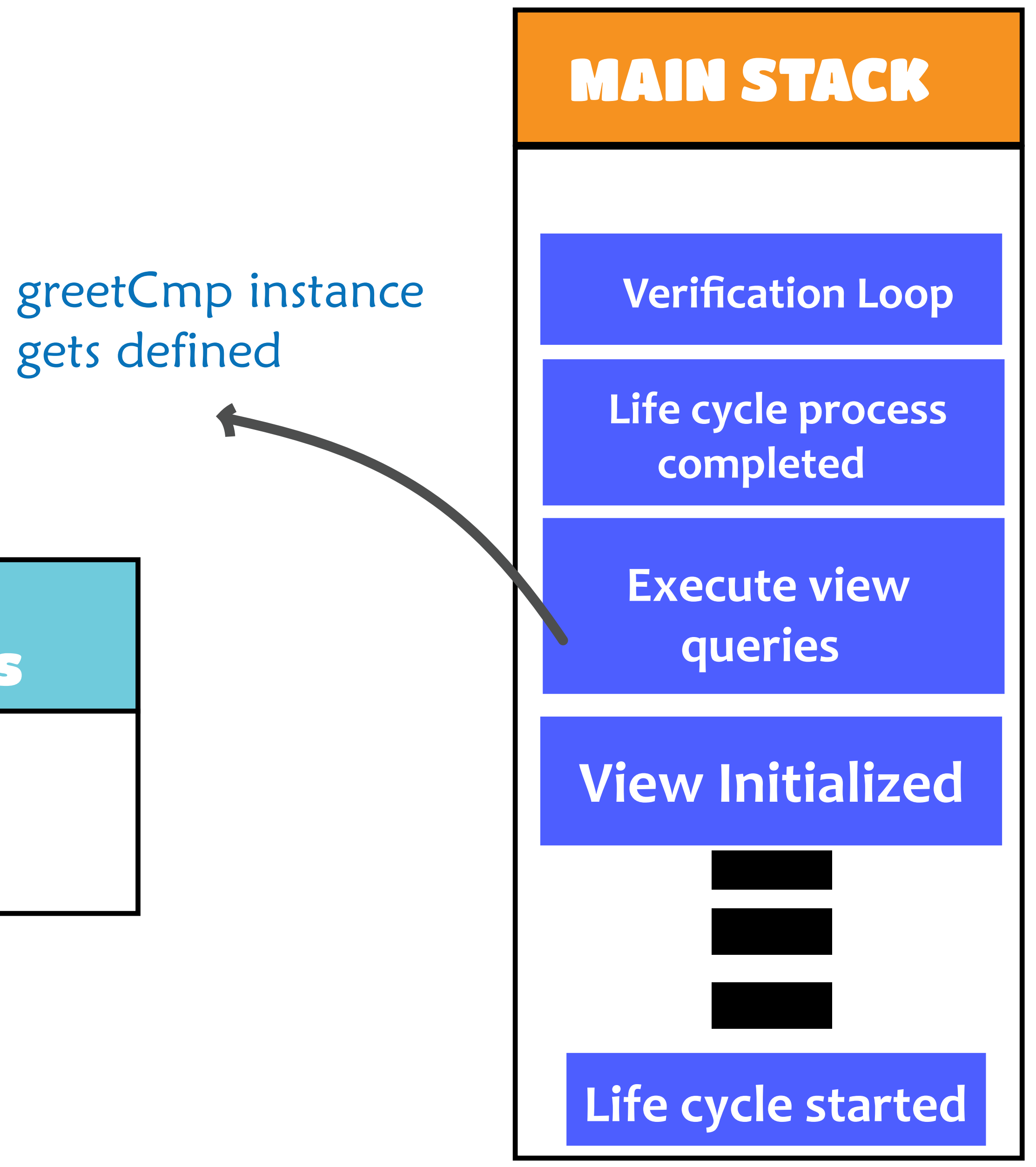


Hello View State (LOCKED)

name: Ajit
greetCmp.isAllowed: null

Hello Execute / Verify Changes

name: Ajit
greetCmp.isAllowed: true



Viewport

Hello
Ajit
For you
Flowers

SLIDE 8 - Child Component Prop Binding

hello.component.html

```
<h1> Hello </h1>
{{ name }}
<p *ngIf="greetCmp.isAllowed"> For you </p>
<app-greet> </app-greet>
```

hello.component.ts

```
@Component({ ... })
class HelloComponent {
  @ViewChild(GreetComponent, { static: true })
  greetCmp: GreetComponent;
  name = ' Ajit ' ;
}
```

Hello View Data

instance (component instance ref)
bindingsCount : 2
name (binding)
greetCmp.isAllowed (binding)
app-greet (component)

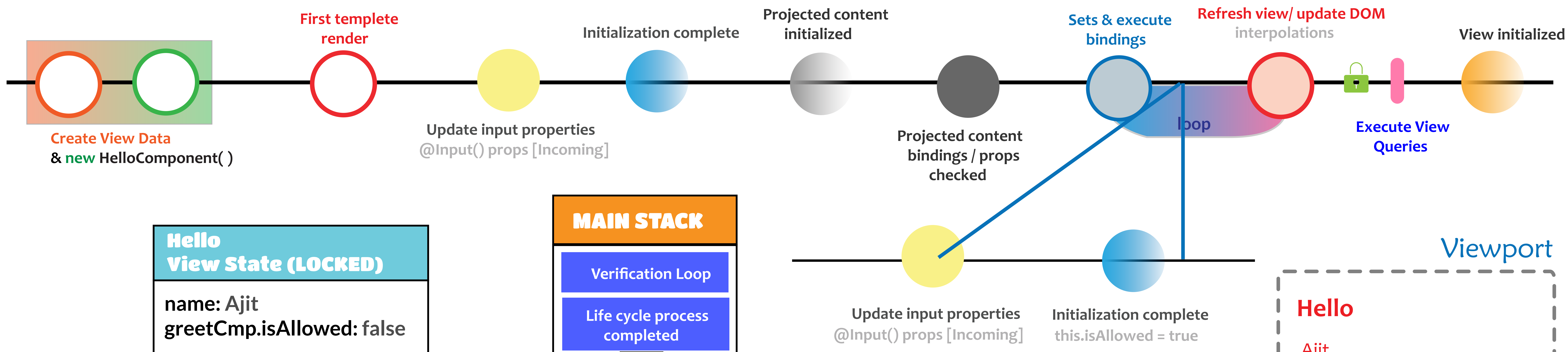
Hello Component instance

greetCmp : GreetCmp (static resolve)
name : Ajit

(class instance)

Greet Component instance

isAllowed: true ~~false~~ (ngOnInit)



Hello View State (LOCKED)

name: Ajit
greetCmp.isAllowed: false

Hello Execute / Verify Changes

name: Ajit
greetCmp.isAllowed: true

MAIN STACK

Verification Loop
Life cycle process completed
app-greet initialization
Static resolve child queries
Life cycle started

Viewport

Hello
Ajit
For you
Flowers

According to Hello View Data
1. The name binding executed
2. The greetCmp.isAllowed binding executed
3. The app-greet initialization
4. The app-greet initialization causes isAllowed = true

hello.component.html

```html
<h1> Hello </h1>
{{ name }}

<app-greet> </app-greet>

<p *ngIf=”greetCmp.isAllowed”> For you </p>
```

hello.component.ts

```typescript
@Component({ ... })
class HelloComponent {
  @ViewChild(GreetComponent, { static: true })
   greetCmp: GreetComponent;
   name = ‘ Ajit ’ ;
}
```

Hello View Data

instance (component instance ref)
bindingsCount : 2
name (binding)
app-greet (component)
greetCmp.isAllowed (binding)

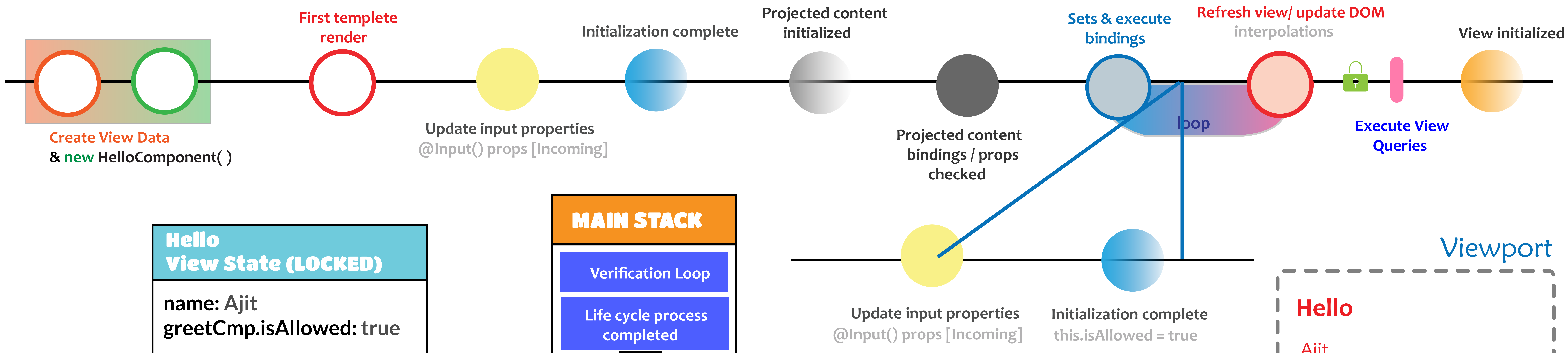
Hello Component instance

greetCmp : GreetCmp (static resolve)
name : Ajit

(class instance)

Greet Component instance

isAllowed: true ~~false~~ (ngOnInit)



Hello View State (LOCKED)

name: Ajit
greetCmp.isAllowed: true

Hello Execute / Verify Changes

name: Ajit
greetCmp.isAllowed: true

MAIN STACK

- Verification Loop
- Life cycle process completed
- app-greet initialization
- Static resolve child queries
- Life cycle started

Viewport

Hello

Ajit

Flowers

For you

According to Hello View Data
1. The name binding executed
2. The app-greet initialization
3. The app-greet initialization causes isAllowed = true
4. The greetCmp.isAllowed binding executed

hello.component.html

```html
<h1> Hello </h1>
{{ name }}
<p *ngIf="greetCmp.isAllowed"> For you </p>
<app-greet> </app-greet>
```

hello.component.ts

```typescript
@Component({ ... })
class HelloComponent {
   @ViewChild(GreetComponent, { static: true })
    greetCmp: GreetComponent;
    name =  ' Ajit ' ;
}
```

Hello View Data

instance (component instance ref)
bindingsCount : 2
name (binding)
greetCmp.isAllowed (binding)
app-greet (component)

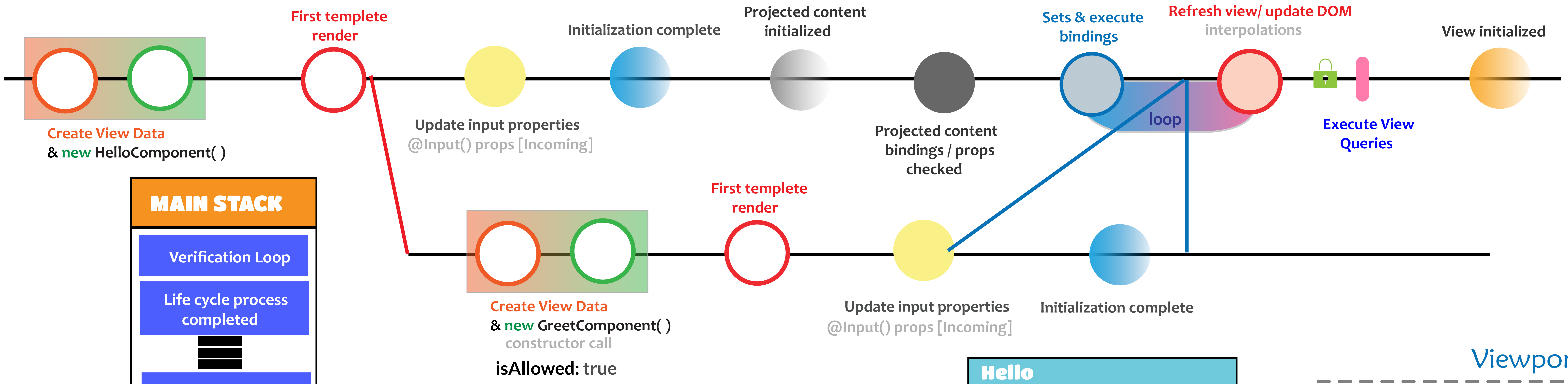
Hello Component instance

greetCmp : GreetCmp (static resolve)
name : Ajit

(class instance)

Greet Component instance

isAllowed: true (constructor)



MAIN STACK

Verification Loop
Life cycle process completed
app-greet view Data
Static resolve child queries
Life cycle started

1. The app-hello view data
2. The app-greet view data, constrcutor call causes isAllowed = true
3. The name binding executed
4. The greetCmp.isAllowed binding executed

Hello View State (LOCKED)

name: Ajit
greetCmp.isAllowed: true

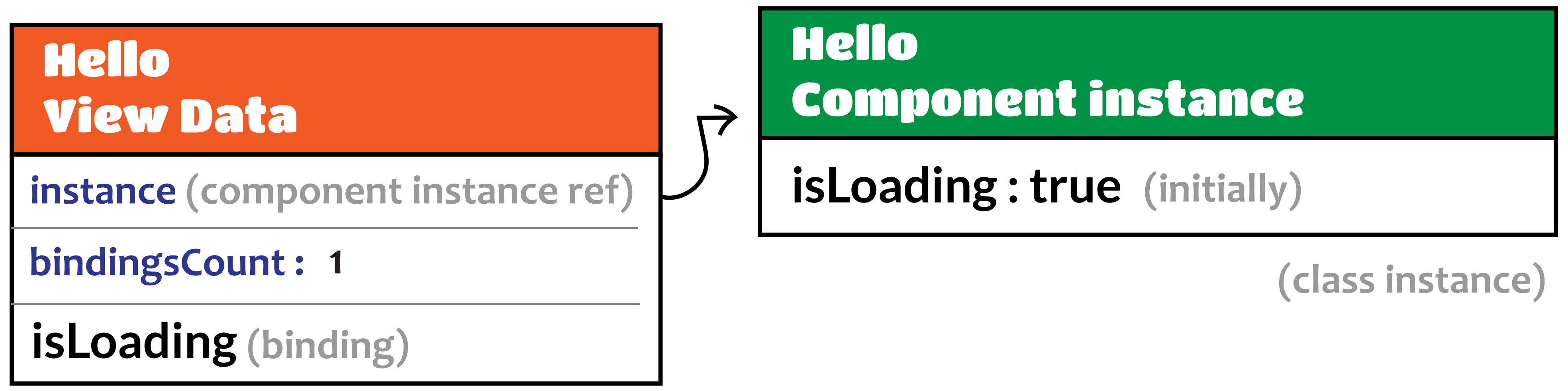
Hello Execute / Verify Changes

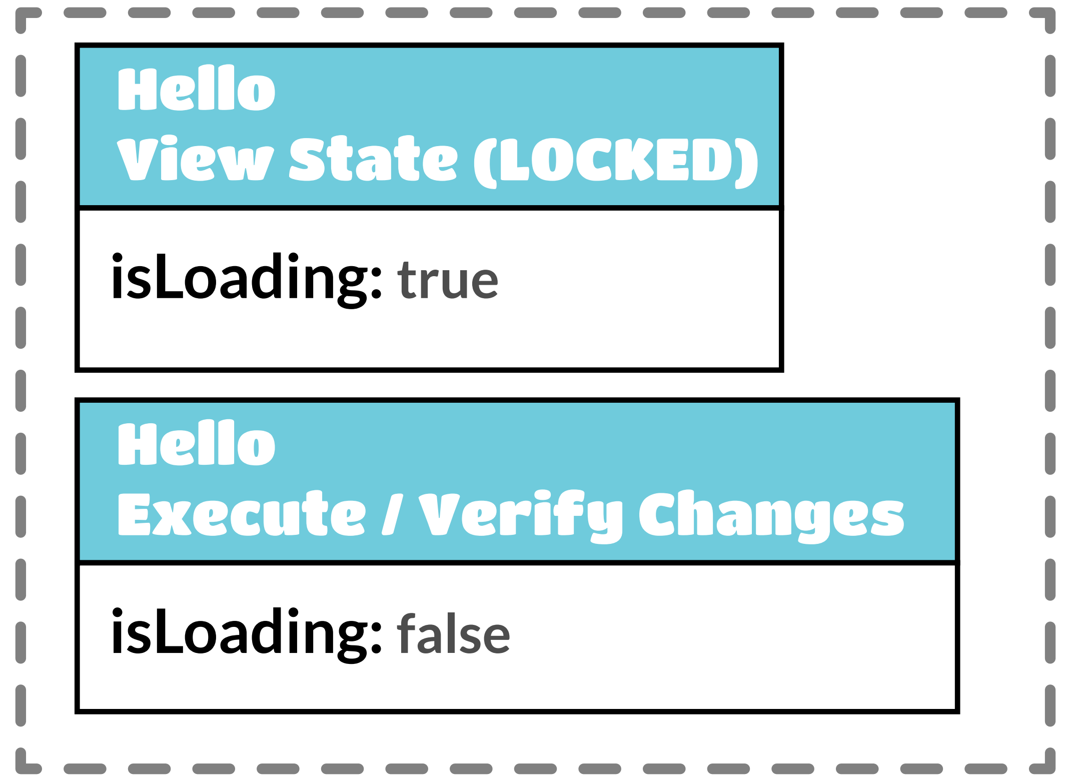
name: Ajit
greetCmp.isAllowed: true

Viewport

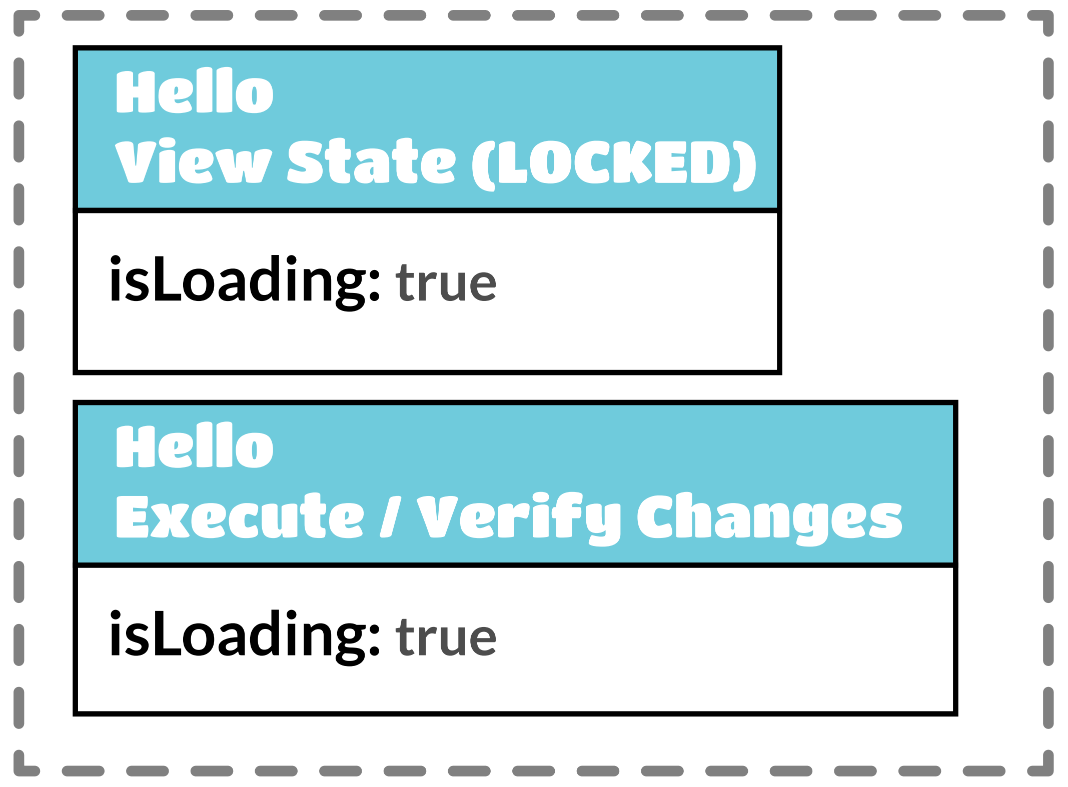
Hello
Ajit
For you
Flowers

SLIDE 11 - Constructor & Bindings Exectution



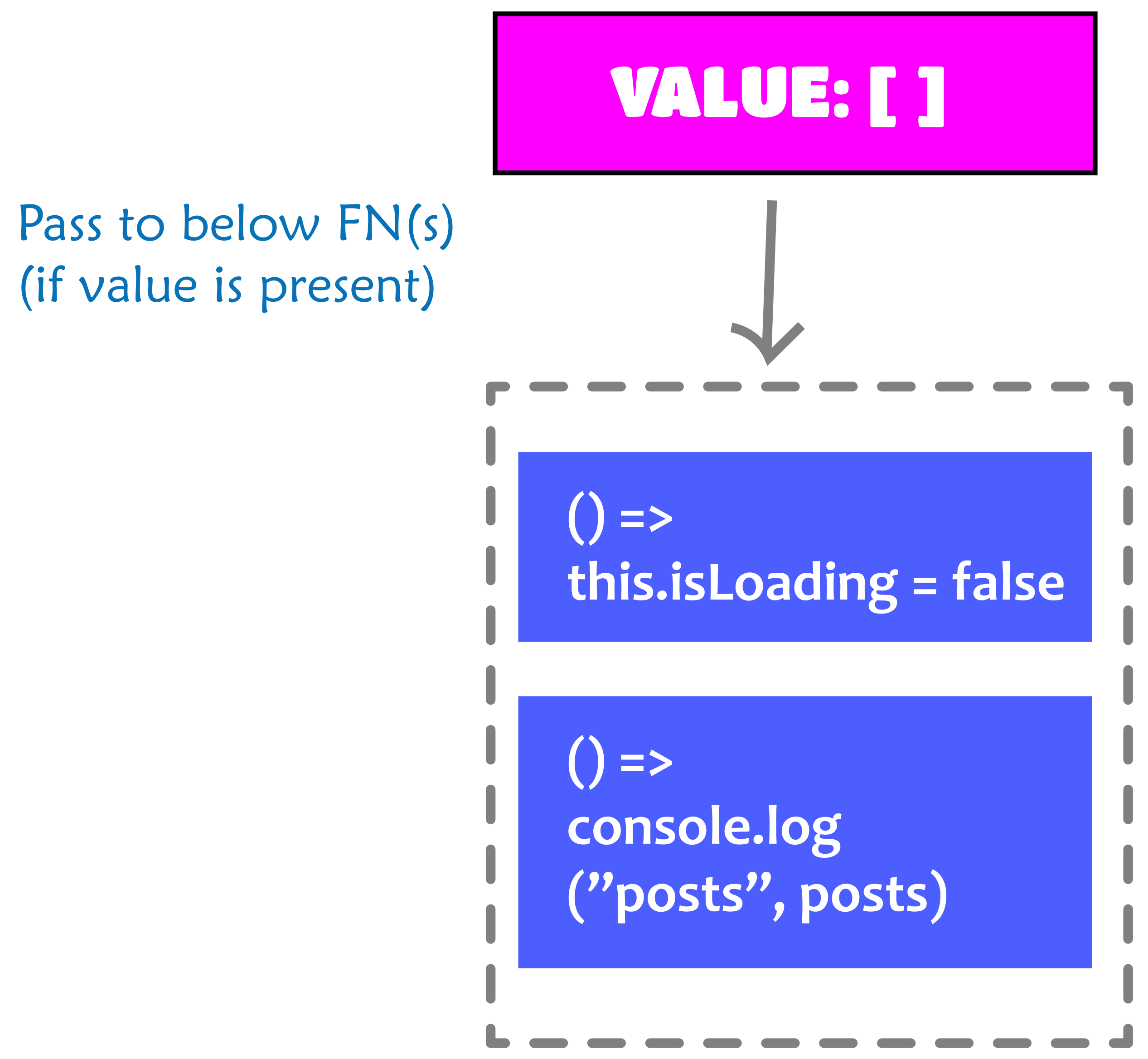


startWith case

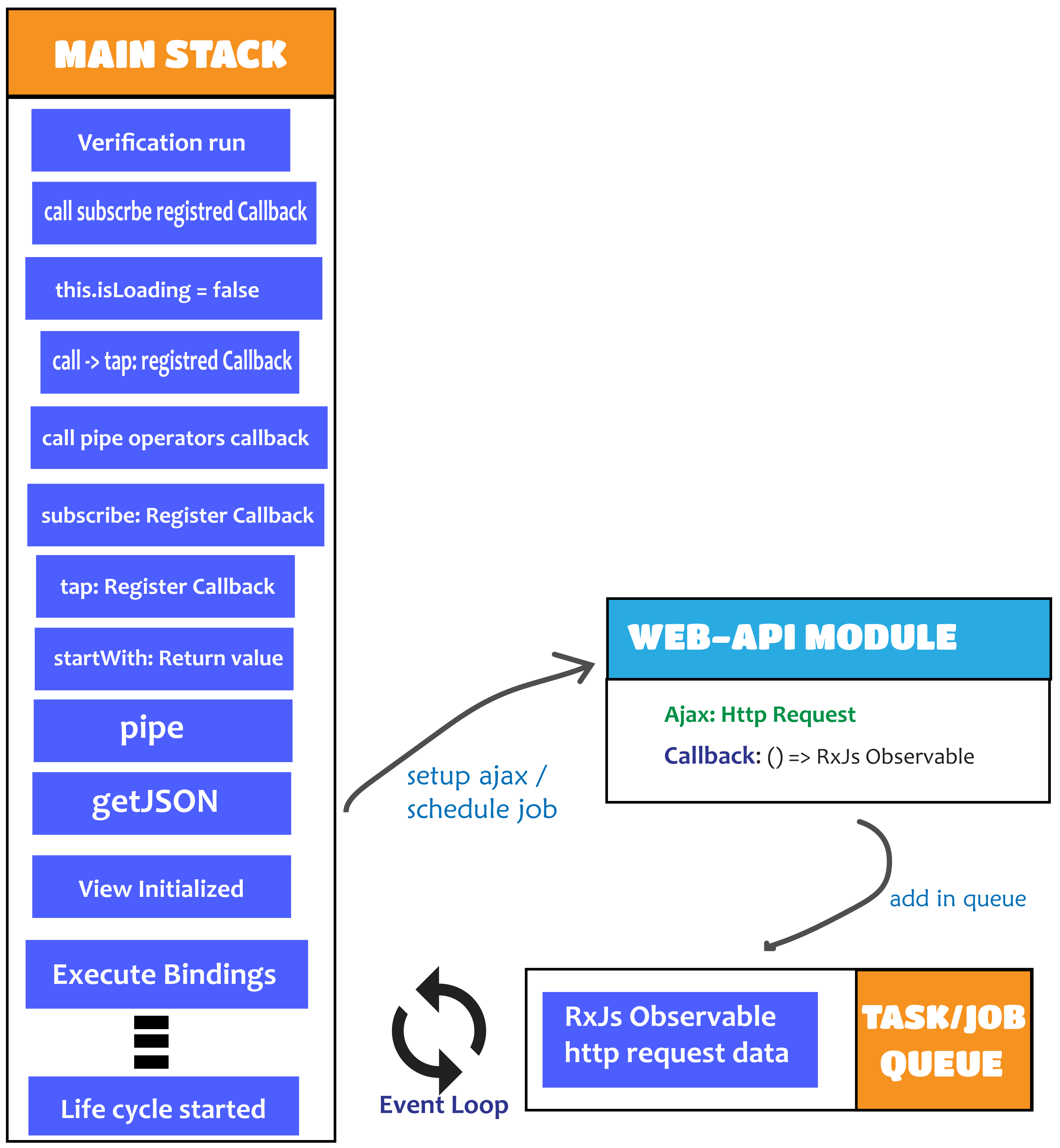


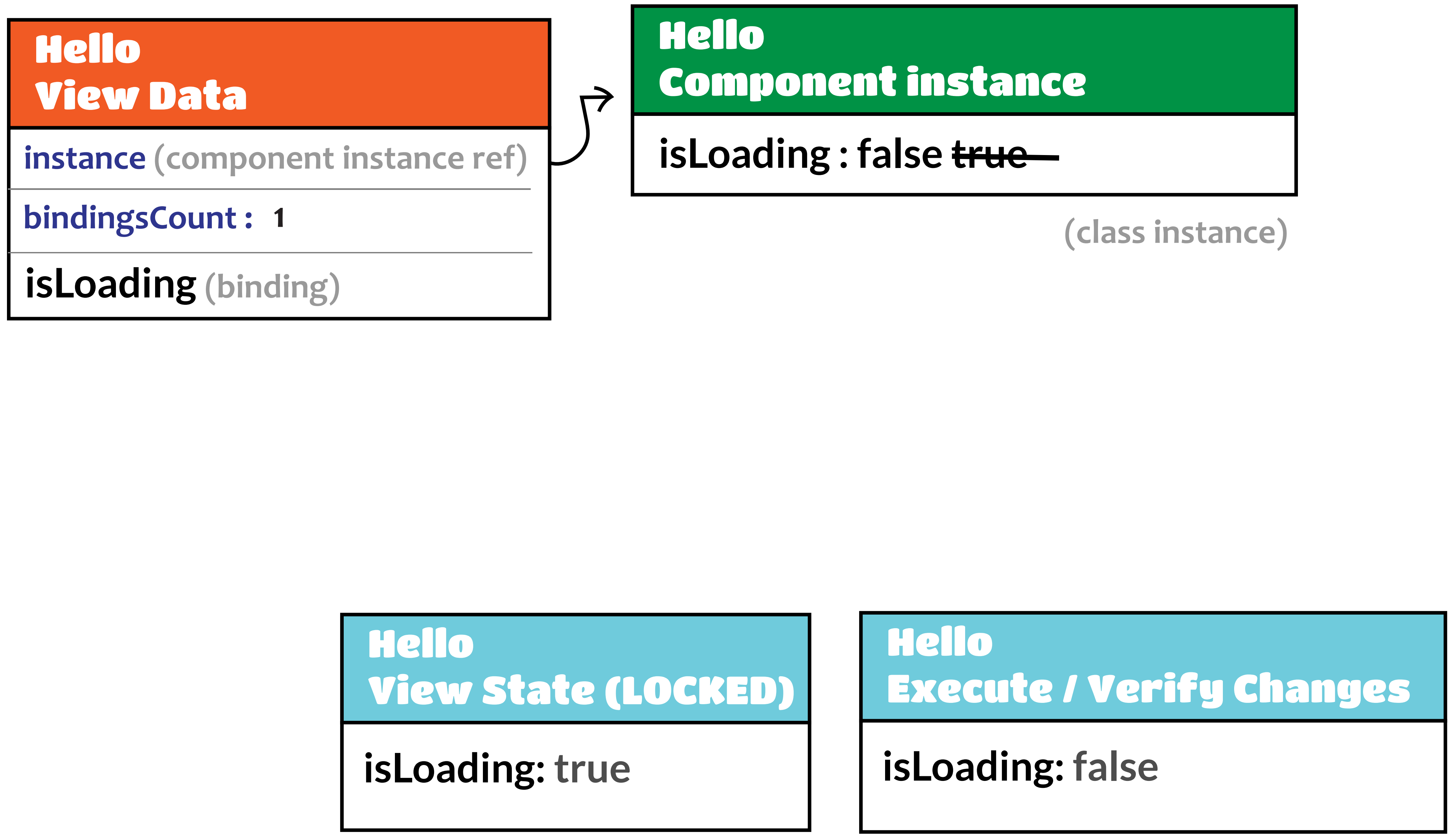
without startWith case

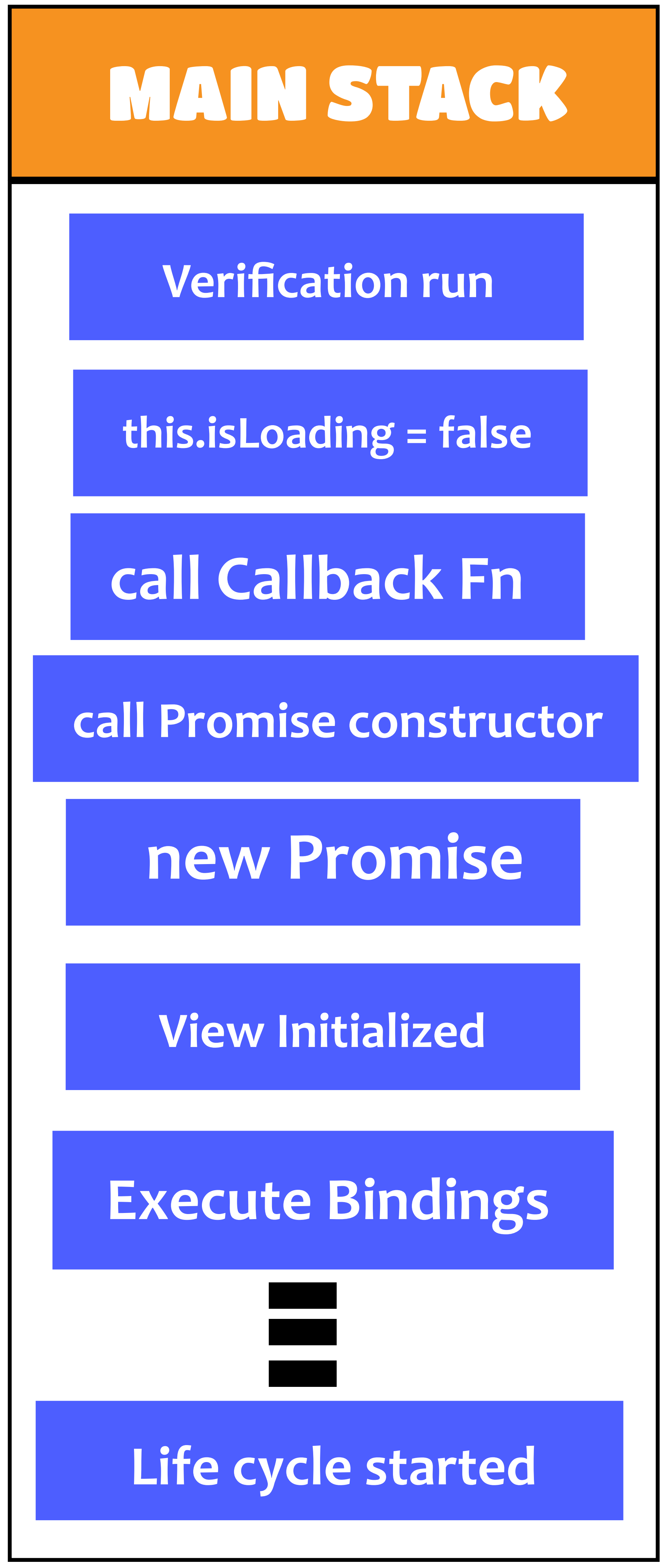
1. Call operators and store values (if any)
2. [startWith: V, tap: Fn]
3. If any operator has return a value, store it
4. Call Subscribe and **register** callabck Fn
5. If any value V already present
    a. Call pipe operators **registred** callbacks [tap: Fn]
    b. Call **registered** callback Fn (subscriber)
   else
    Wait for value (Next tick), then repeat step 5



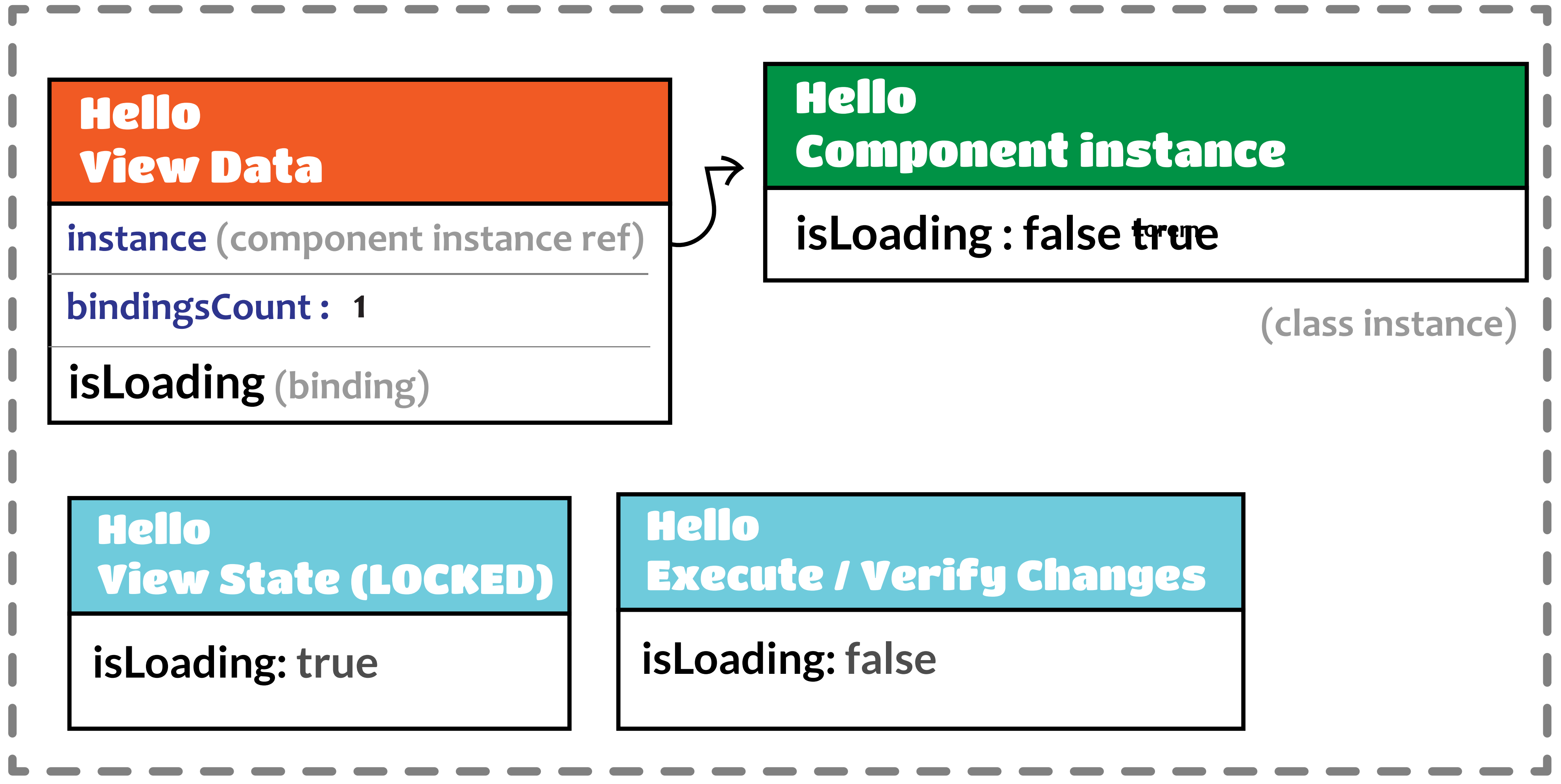
RxJs Registered callbacks



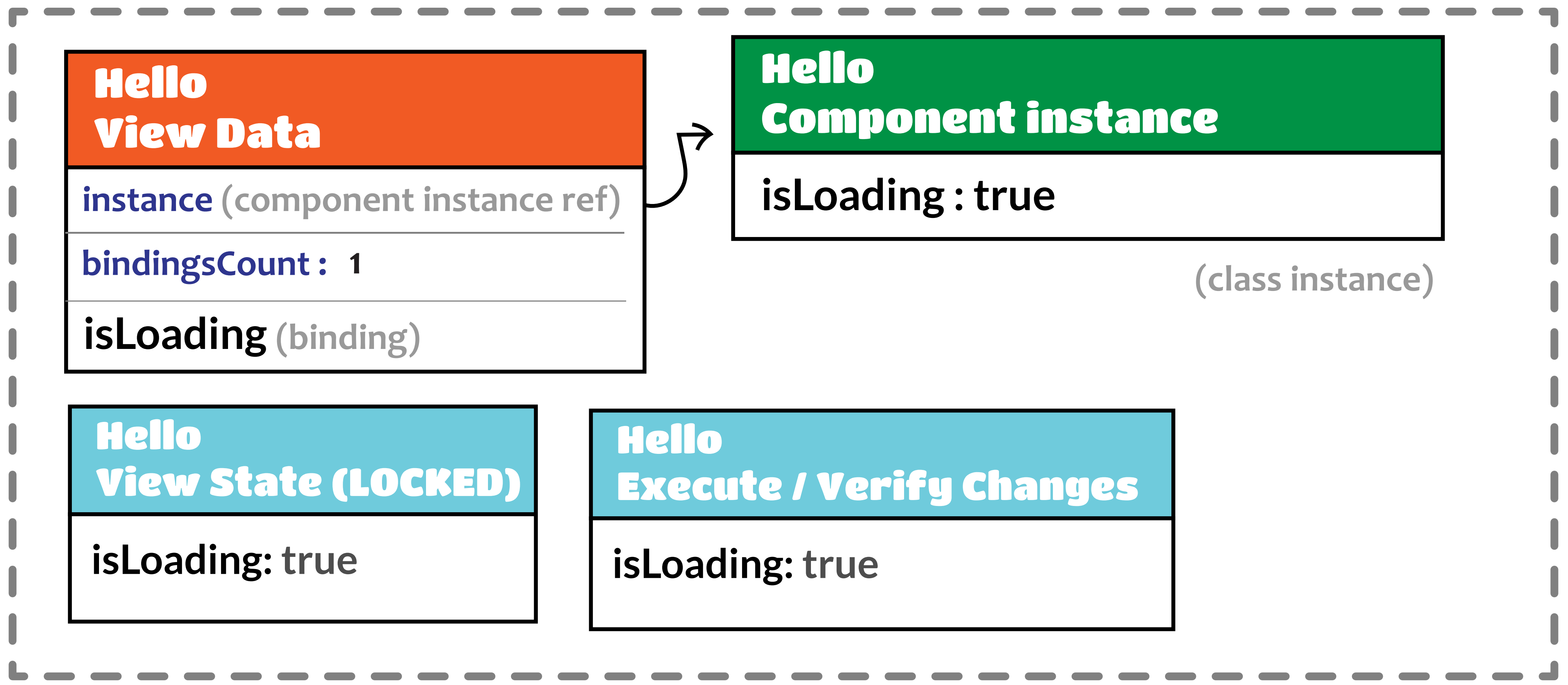




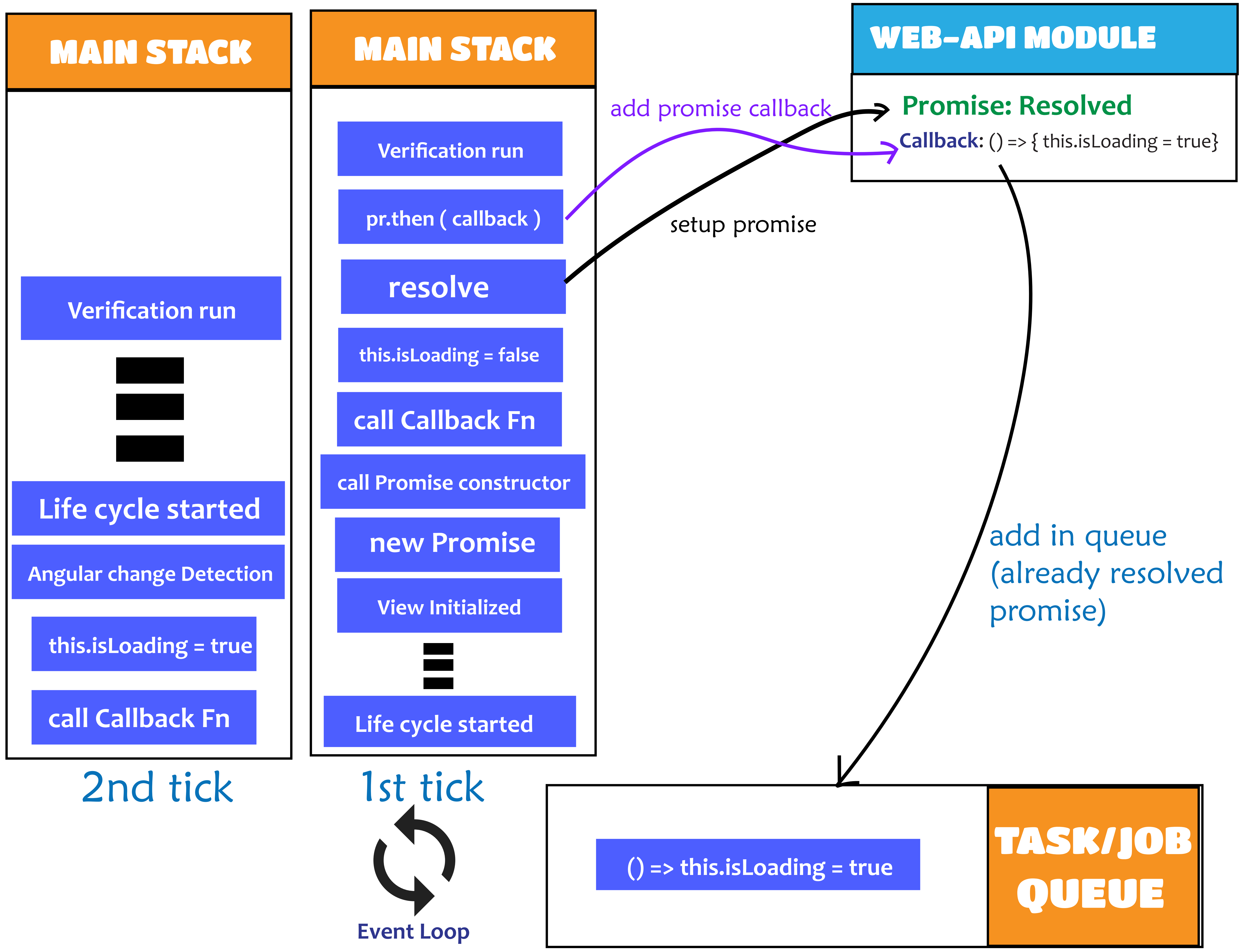
1. Call **Promise constructor** with callback Fn
2. Call **callback** Fn
3. If **resolve** is called, call the **then** registred callback
4. If **reject** is called, call the **catch** registered callabck



1st tick



2nd tick



1. Call **Promise constructor** with callback Fn
2. Call **callback** Fn
3. If **resolve** is called, call the **then** registred callback
4. If **reject** is called, call the **catch** registered callabck